DBSCAN

- DBSCAN refers to Density-based spatial clustering of applications with noise
- DBSCAN works fairly well with large data and can handle noise and outliers very efficiently.

Density and Dense Region

- At a certain point P, density at point P is the number of points within a hypersphere centered at P with a radius of $epsilon$
- Now, consider any region around the point P within eps radius, if there are more data points than $minpts$, we call the region a **Dense** region.
- For example, let's say we have eps=1 and $minpts$=10. Consider two points P_1 and P_2, both with a radius of eps
    - Suppose there are 20 points around point P_1, and only 6 points around point $P_{2,}$ within the radius of eps, then we say the region around point P_1 is dense and the region around point P_2 as non-dense.

Min Points($minpts$) and Epsilon(eps)

- $minpts$ is the minimum number of points that we need in a hypersphere around point P with the radius of eps for considering the region as a **Dense** region.
- $minpts$ acts like a certain threshold and eps is the radius of the hypersphere

Core Point

- If a point P has points $\geq minpts$ within the radius of eps, then P is a core point.
- This also implies that point P has a dense region around it

Border Point

- A point P can be defined as a border point if:
    - P is not a core point
    - Point P lies in the **neighborhood** of point Q such that point Q is a core point

Neighborhood

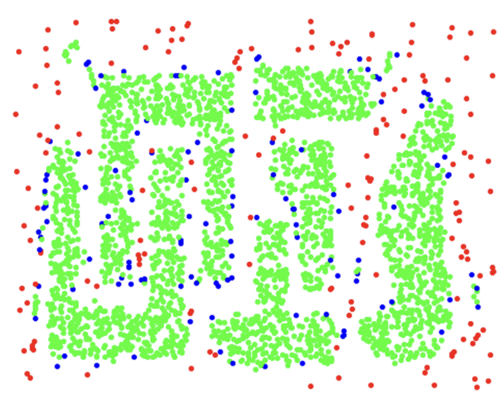
- A point P is said to be in the neighborhood of point Q if the distance between point P and Q is less than eps value; i.e. $dist(P,Q) \leq eps$

Noise Point

- It is a point that is **neither a core point nor a border point.**
- Suppose around core point P, a border point Q, and a point R which is in a non-dense region, the point R is said to be a noise point

- One thing to understand is that, when using DBSCAN, we fix two things:
    1. Min Points
    2. Epsilon.
- By fixing these hyperparameters, we get core points, border points, and noise points as well

Original Points



Point types: core, border and noise

Density Edges and Density Connected Points

- If points P and Q are two core points and the distance between point P and Q is less than or equal to eps value, then an edge between point P and Q is known as a **density edge.**

- Points P and Q can be said as density-connected points;
    - if both points are core points
    - if there exist other density edges connecting the points P and Q

- Example: Imagine we have two core points, point P, and Q, and there are other core points connecting point P with point Q; say P_1,P_2,....Pn, where the distance between each point P_1,P_2,....Pn is less than eps
    - Then point P and point Q are said to be density-connected points.

DBSCAN Algorithm

Step-1:

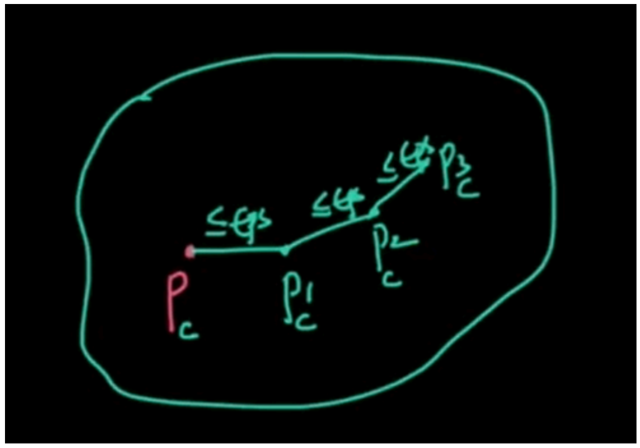
- For each point, xi that belongs to the dataset D, label it as either core point, border point, or noise point.
- The time complexity of this step would be $O(n*logN)$

Step-2:

- Remove all the noise points from the dataset
- The time complexity of this step would be $O(n)$
- This is a noise-removal step

Step-3:

- For each core point P that is not yet assigned to any clustered:
    - create a new cluster with point P
    - Add all points that are density connected to point P, to the P's cluster
- To understand this with an example, Consider a core point P and there are three core points P_1,P_2 and P_3 which are density connected.
- Then, we group all three points in the cluster of point P
- The time complexity of this step would be $O(n*logN)$



Step-4:

- For each border point, we assign it to the nearest core points' cluster.
  - For example, if we have a cluster having core points P_1,P_2....P_9, and a border point P_{10} which is near the cluster.
  - We merge border point P_{10}, into the cluster of core points P_1,P_2....P_9
- The time complexity of this step would be $O(n)*logN$

Adjusting MinPoints

- The value of $minpts$ should be greater than or equal to d+1; where d is the dimensionality of the data
  - a lot of libraries use the value of $minpt$s approximately equal to 2$*d$
- Given an epsilon value, if the dataset is noisy, we pick larger $minpts$

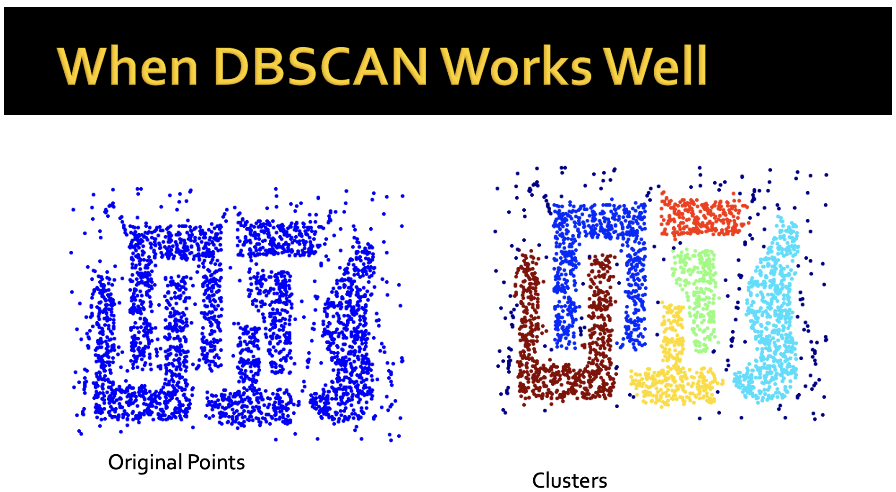
Adjusting Epsilon

- Let's assume we've fixed the value of $minpts$ = 4.
- **Step 1:**
  - for every point xi in the dataset, we compute a distance di
  - di refers to the distance from xi to xi's 4th nearest neighbor (because we've set $minpts$ = 4)
- **Step 2:**
  - Sort the values of di's and plot them. You'll notice that the distance will increase gradually and then suddenly, at a certain point, the value of distance will get boosted

- So, the index at which the value of di distance got boosted will be used as the value of eps
- The indices having higher values of di's will be outliers

Advantages of DBSCAN

- It's resistant to noise
- Can handle clusters of different shapes and sizes.
- It doesn't require one to specify the number of clusters a priori.
- It requires only two parameters: MinPts and Epsilon.



Limitations of DBSCAN

- Even with a small change in the hyperparameters, we can get a completely different type of cluster. So, it's quite sensitive to the choice of hyperparameters.
- Cannot handle varying densities and data with higher dimensions.

Anomaly Detection

What is an Anomaly?

- Anomaly is synonymous with an outlier. These terms are often interchanged and may be called Novelty depending on the context.

What's the difference?

- Anomaly means something which is not a part of the normal behavior
- Novelty means something unique, or something that you haven't seen before(novel)

1. Distribution Based

- The simplest way to detect an outlier would be to use distribution parameters (mean and standard deviation).

Problem with this approach:

- While we know the distribution, the parameter estimates of the distribution are often corrupted by the noise/outlier
- Hence, we need to robustly estimate the parameters of the distribution. => RANSAC

2. Random Sample Consensus (RANSAC)

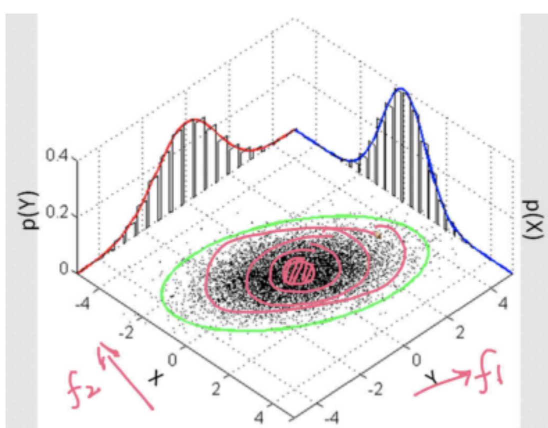
- Imagine a dataset X with datapoints having parameters μ and σ. Let's call them collectively θ
- Steps:
    - Sample a subset of points from the dataset ($n^{'}$). We consider this point as an inlier.
    - Now, compute a model that estimates the parameters of the sampled points.
    - Score the model which indicates how many points will support the model.
- We repeat these three steps iteratively and then select the model best supported by the data, which then tells which points are inliers and which are outliers.

Extending the idea to higher dimensions

- Till now, we assumed that we have only a single feature X which follows Gaussian Distribution.
- Now, imagine we have d-dimensional data where each point x_i Є R^d and the data is not labeled.
- If we know the data points x_is follow multivariate Gaussian distribution(unimodal),
  - then X follows normal distribution; X ~ ($\vec{u}$, Σ), where is $\vec{u}$ mean vector and Σ is a covariance matrix
- Here we'll consider ($\vec{u}$, Σ) as θ
- In GMMs, in multidimensional space, the shape of Gaussian was similar to a hill where the density of the points was highest in the middle contour, and it keeps getting low as we move away from the center
- In this case, too, RANSAC can be applied. Farther away from centroid, we'll know that it is an outlier.

3. Elliptic Envelope

- We know that a Unimodal Multivariate Gaussian Distribution on a single plane will look like ellipses if visualized on a plane. This idea can be extended to find out an outlier



- Given some data X where x_is $\in R^d$ and X follows Normal Distribution being unimodal, Elliptical Envelope robustly estimates the parameters ($\vec{u}$, Σ).

- The term robustly means without getting impacted by outliers

- We remove the points that are outliers which are very far away from the centroid

Disadvantages:

- It cannot be used for non-unimodal data
- It is specifically for multivariate Gaussians
- If the data fails to meet the assumptions of unimodal and multivariate Gaussian, the whole thing crashes.